AN IIOP ARCHITECTURE FOR WEB-ENABLED PHYSIOLOGICAL MODELS

Shixin Zhang and C. Forbes Dewey, Jr.
Department of Mechanical Engineering, Massachusetts Institute of Technology,
Cambridge, MA 02139 USA

Abstract- **This paper develops a specific information architecture to serve complex physiological information models and a means of delivering these models in a manner that allows interactive and distributed use. By redesign of existing models for distributed use, the IIOP (Internet Inter-ORB Protocol) architecture provides general access across the Internet; the methods are replicable with many different types of physiological models that produce a variety of results.**

This paper defines and explains the complete architecture for the user interface, the model encapsulation, and the communication layer between the client and server by developing a general example. Using the equivalent of interactive browsers to access remote models and display the results, the IIOP architecture is built up using platform-independent technology such as CORBA, Java and XML. The existing physiological models are first encapsulated by a suitable software language respect to the legacy models. Then CORBA IDL-XML interfaces are built accordingly as a broker interface connecting user interfaces to encapsulating interfaces. Therefore, the standard user interfaces on the browsers are easily built to access these models through the CORBA ORB and the encapsulating interfaces. This interface software is capable of interpreting and displaying very high-level descriptors and model output such that the amount of data required to be transmitted over the Internet is reduced.

1. INTRODUCTION

Models of human physiology are at the heart of clinical prediction and the teaching process in biomedical engineering. These Models, developed over past 30 years, have become not only quantitative but also computationally intensive. The power of these models to illustrate and predict clinically relevant physiology has also become enormously important in health care delivery. But in most current embodiments, the user must sit at the same computer upon which the model runs or use X-window to simulate the environment. What is needed is a means of delivering these models in a manner that allows interactive and distributed use, such as Internet learning and distance education.

The comprehensiveness and complexity of physiological models are increasing for two reasons. First, the questions being asked are much more demanding. Second, the level of interdependence of the component models has increased dramatically in order to answer complex physiological questions. Frequently the variety of knowledge required to answer a single physiological question exceeds the expertise of any individual group of researchers, and one is required to interface different models from many different sources with different local design rules. [1]

An information architecture that will serve complex physiological information models must support interactive and distributed use. This may require the redesign of the existing models for distributed use. The software methods should be usable with many different types of physiological models that produce a variety of results: graphical, numerical, and images (including moving images). The software developed should preferably be publicly available.

This need can be met by a web-based architecture that uses the equivalent of interactive browsers such as Netscape and Microsoft Explorer to access remote models and display the results. This access route can be built up using platform-independent technology such as HTML, XML, Java and CORBA. In a nutshell, the existing physiological modeling must first be encapsulated by a suitable software language, such as Java and C++ with respect to the legacy language and application. Then CORBA IDL-XML interfaces are built accordingly as a broker interface connecting user interfaces to encapsulating interfaces. Therefore, the standard user interfaces on the browsers can be easily built to access these models through the CORBA ORB and the encapsulating interfaces. This interface software is capable of interpreting and displaying very high-level descriptors and model output such that the amount of data required to be transmitted over the Internet is reduced.

The web-enabled information architecture and an example are given in detail in this paper.

2. PHYSIOLOGICAL MODEL EXAMPLE

We have centered our project on many specific use cases where there are existing models that either use X-window interface or have an interface that is extremely limited. Our experience is that having concrete examples that range across the spectrum of possible applications is the best way to insure that the software architecture and its implementation will meet the general needs of users.

Limited by the paper space, this paper only uses one general example to illustrate the IIOP architecture we developed. The example is "A Systemic Circulation Lumped-Parameter Model: Quantitative Physiology: Organ Transport Systems", developed by Professor Roger D. Kamm and his colleagues at MIT. In this model, a non-linear, distributed model of the arterial system and heart is developed which is based upon a numerical solution of the one-dimensional equations of motion in a geometrically accurate branching network of the arterial system. Inputs to the model are hemodynamic parameters such as Systemic

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
25 Oct 2001	N/A	-

Title and Subtitle An IIOP Architecture for Web-Enabled Physiological Models	**Contract Number**
	Grant Number
	Program Element Number
Author(s)	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Department of Mechanical Engineering Massachusetts Institute of Technology Cambridge, MA 02139	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es) US Army Research, Development & Standardization Group (UK) PSC 802 Box 15 FPO AE 09499-1500	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)

Distribution/Availability Statement
Approved for public release, distribution unlimited

Supplementary Notes
Papers from 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, October 25-28, 2001, held in Istanbul, Turkey. See also ADM001351 for entire conference on cd-rom., The original document contains color images.

Abstract

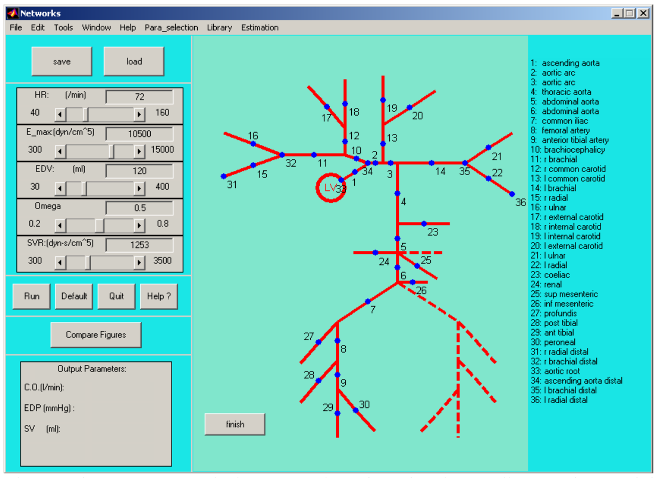
Subject Terms

Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU

Number of Pages
4

Venous Resistance (SVR) which are critical parameters to evaluate the cardiovascular system but can usually be gained only through invasive measurements. Outputs of the model are pressure and flow waveforms at all locations of arterial system. [2]

This computational model was originally written using the proprietary mathematical package MATLAB and worked only under the X-windows system (Fig. 1). We have created a general interface for the model that can be executed under conventional web browser technology on all computers, and changed the model so it can be executed using a non-proprietary compiled and freely distributable executable. The connection software between the model and the browser will meet the same design rules as all the other models that we support.


Networks
File Edit Tools Window Help Para_selection Library Estimation
save
load
Run
Default
Quit
Help ?
Compare Figures
finish
LV
33

HR: (/min) 72 — 40 … 160
E_max:(dyn/cm^5) 10500 — 300 … 15000
EDV: (ml) 120 — 30 … 400
Omega 0.5 — 0.2 … 0.8
SVR:(dyn-s/cm^5) 1253 — 300 … 3500

Output Parameters:
C.O.(l/min):
EDP (mmHg) :
SV (ml):

1: ascending aorta
2: aortic arc
3: aortic arc
4: thoracic aorta
5: abdominal aorta
6: abdominal aorta
7: common iliac
8: femoral artery
9: anterior tibial artery
10: brachiocephalicy
11: r brachial
12: r common carotid
13: l common carotid
14: l brachial
15: r radial
16: r ulnar
17: r external carotid
18: r internal carotid
19: l internal carotid
20: l external carotid
21: l ulnar
22: l radial
23: coeliac
24: renal
25: sup mesenteric
26: inf mesenteric
27: profundis
28: post tibial
29: ant tibial
30: peroneal
31: r radial distal
32: r brachial distal
33: aortic root
34: ascending aorta distal
35: l brachial distal
36: l radial distal

Fig. 1: The current X-windows user interface for the cardiovascular model based on MATLAB proprietary technology[2]

where:

HR = Heart Rate (beats/min);
EDV = End Diastolic Volume (ml);
SVR = Systemic Vascular Resistance (dyn/cm^{5}sec);
ELV_Max = Left Ventricle Elasticity (Contractility) (dyn/cm^5);
Omega = Ejection Period (Time$_{systole}$/Time$_{total}$);
CO = Cardiac Output (L/min);
SV = Stroke Volume (ml)
EDP = End diastolic pressure (mmHg)

3. IIOP ARCHITECTURE

As we discussed in the previous sections, our needs can be met using a web-based architecture that uses the equivalent of interactive browsers to access remote models and display results. This access route can be built up using platform-independent language and technology such as JAVA, CORBA and XML. The CORBA/IIOP architecture lets distributed objects communicate by use of object request brokers, or ORBs. CORBA - which is, in itself, a complete distributed object platform - extends the reach of Java application across networks, languages, component boundaries, and operating systems [3]. Java begins where CORBA leaves off. CORBA deals with network transparency and Java deals with implementation transparency. The class of CORBA is neither part of an operating system nor an application, but is used to link together the various parts of a distributed application spread across geographically separated computers. It is specifically designed to be the glue that binds disparate programming technologies together. It does not exist as a point in the programming space; by design, it occupies the spaces between the peaks representing individual languages [4]. From our standpoint, one of CORBA's most valuable traits is that it is designed to be language and platform independent. The cross-platform and cross-language nature of CORBA is definitely its powerful feature. For our purposes, where models are naturally distributed and are implemented with a variety of different languages, CORBA, we'll show, is a powerful implementation tool. When a Java client uses CORBA technology to communicate with a C++ object, for example, both the C++ programmer are the Java programmer work completely within their respective language environments. The CORBA ORB presents the Java client with a Java stub interface and the C++ programmer with a C++ skeleton interface. CORBA takes care of the cross-language issues automatically. [4]

The information system architecture we developed is illustrated in Fig. 2. First, the client user loads the Java applet to his Web browser using HTTP protocol; then the applet sends a request to the CORBA-ORB server program through IIOP protocol. The server program communicates with the encapsulated models and then sends back the results from the models to the Java applet on the client side through IIOP or HTTP. We find it is easy to build up an integrated system by using CORBA-ORB technology. The same ORB can be used to communicate with databases and other model objects without any change of the whole architecture. Another advantage of this architecture is that we can replace the legacy model with a new model (for example, a library), without any change of the architecture. The client user will never realize any difference.

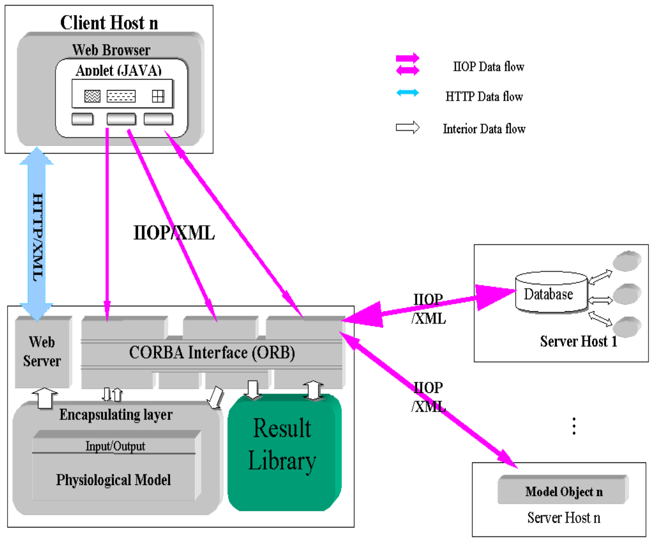


Fig. 2: Architecture design: IIOP Web application architecture

In this architecture, we first need to build up an encapsulating layer for the existing model with a suitable technology, according to the legacy language and application situation. Second, since we use IIOP protocol, we need to define CORBA IDL interfaces for each model as a broker interface which connects the client Java applet to the encapsulating layers. Third, we need to build a standard user interface that will be loaded on the browser.

Implementation of the route of Fig. 2 involves the following activities usually:

1. We need to perform modest surgery on the existing model, because it usually has its special GUI that we can't deliver directly to the Internet. So we need to understand which input variables and output variables the model uses and where they are. Then we cut off the GUI section and keep the computation section separately.
2. Design and implement the encapsulating layer's classes for the model. This is a key step and there are many techniques for wrapper coding. This is discussed in the next paragraph.
3. According to the input and output variables and structure, we need to design and define a CORBA IDL interface.
4. Automatically generate the stub and skeleton classes and interfaces using the ORB's IDL compiler.
5. Design and implement the server's classes that implement the functionality defined by the server's IDL interface and communicate with the encapsulating sections, including the server's initialization code and the intersection of the server's implementation classes, meanwhile linking with the skeleton classes produced by the IDL compiler.
6. Design and implement the client Java applet based on the functionality of the original model GUI, and any other functionality necessary but not visible to the client applets and applications. We can write a Java applet with the GUI style of original model in order to keep the user interface unchanged, or replace the orginal functionality with a new paradigm.
7. Define XML elements to encapsulate the structure data during the transmitting process, in order to communicate efficiently with applet, databases and other models. This means encoding and decoding of all object classes as shown in Fig. 3.

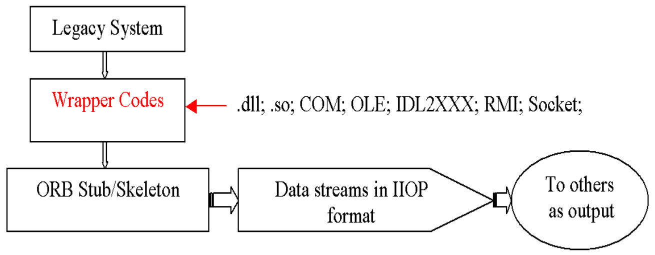

Legacy System
Wrapper Codes
.dll; .so; COM; OLE; IDL2XXX; RMI; Socket;
ORB Stub/Skeleton
Data streams in IIOP format
To others as output

Fig. 3: Encapsulating technology

Fig. 3 indicates that to use IIOP architecture, we need to design the wrapper codes for the legacy system. There are many technology choices for this wrapping:

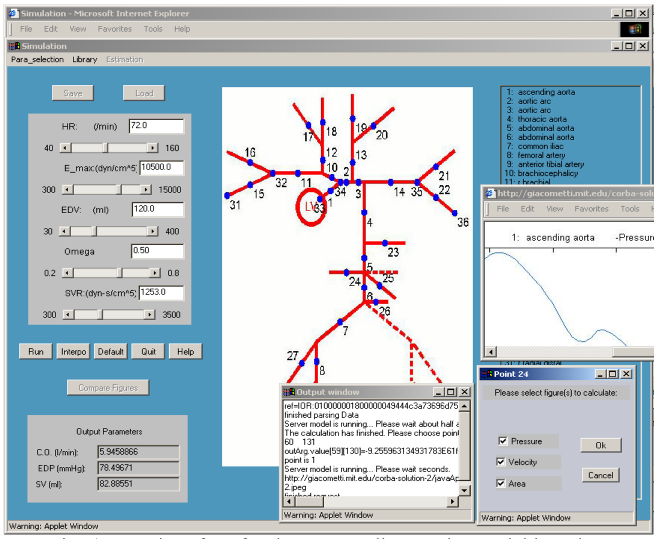
- Use C link libraries provided by original language, such as FORTRAN using "cfortran.h", MATLAB etc;
- Use intersection programs to connect each software component, such as a . DLL or temporary text files;
- Use a third bridge language, such as Tcl/Tk when working with text files and programs;
- Use a translator, for example, IDL2XXX; xxx means the other language, such as PASCAL, C++, JAVA, PERL;
- Use COM/OLE technology;
- Use some special interface, such as Java RMI.

For our physiological example – which used a MATLAB model – there were two open system choices for encapsulation:

- Use the C/C++ language to call the MATLAB engine libraries (C/C++) provided by MATLAB, such as libmx.lib, libeng.lib and libmat.lib;
- Use the MATLAB C/C++ compiler (mcc.exe) to convert MATLAB files (. m files) to C/C++ programs first, then link with the MATLAB C/C++ Math & Graphics Library.

After comparison, we used the first method to encapsulate the model. It cost less work than the second method.

The new model with Web-based architecture gives correct results as Fig. 4 illustrates. We can type the parameters in or use drag bars to choose the values. Clicking the "run" button will call the calculation model and display the results on the left-bottom of the window; clicking the "interpolation" button will get the results from the interpolation table. If you click the numbered points on the diagram of cardiovascular system after calculating, a window with three choices (pressure, velocity and area) will be displayed; then, according to your choice, the program will produce the graphs of the pressure, velocity or area of that point (location), as shown in Fig. 4. There is an "output window" that we use to display the intermediate results.


Simulation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Simulation
Para_selection Library Estimation
Save
Load
Run Interpo Default Quit Help
Compare Figures
http://giacometti.mit.edu/corba-solu
1: ascending aorta -Pressur
Output window
Point 24
Ok
Cancel
Warning: Applet Window

HR: (/min) 72.0
40 160
E_max:(dyn/cm^5) 10500.0
300 15000
EDV: (ml) 120.0
30 400
Omega 0.50
0.2 0.8
SVR:(dyn-s/cm^5) 1253.0
300 3500

Output Parameters
C.O. (l/min): 5.9458866
EDP (mmHg): 78.49671
SV (ml): 82.88551

1: ascending aorta
2: aortic arc
3: aortic arc
4: thoracic aorta
5: abdominal aorta
6: abdominal aorta
7: common iliac
8: femoral artery
9: anterior tibial artery
10: brachiocephalicy
11: r brachial

```
ref=IOR:010000001800000049444c3a73696d75
finished parsing Data
Server model is running... Please wait about half a
The calculation has finished. Please choose point
60   131
outArg.value[59][130]=-9.255963134931783E61f
point is 1
Server model is running... Please wait seconds.
http://giacometti.mit.edu/corba-solution-2/javaAp
2.jpeg
```

Please select figure(s) to calculate:
[✓] Pressure
[✓] Velocity
[✓] Area

Fig. 4: User interface for the new cardiovascular model based on Java applet technology and IIOP architecture; this picture also shows the results response to the remote user request.

After the above work, one serious problem remained: the model was not interactive. The orginal MATLAB model took several hours/case. Each calculation of our compiled model spent from half an hour to several hours on our Pentium III 400MHz machine. Nobody can wait such a long time on the Internet. How do we get the result during a short online time? Our answer is to use a library (database) of precomputed results and an interpolation technique instead of the real calculation (Fig. 5). This solution can save time from several hours to half a minute. The new model using a five-dimensional interpolation runs in seconds without any change in either the client or the network architecture.

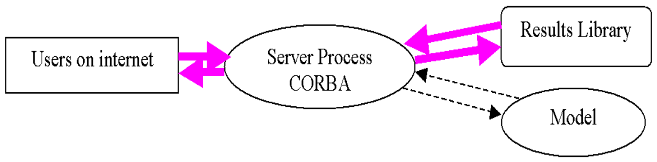


Fig. 5: The performance solution: rip out the existing MATLAB model and replace it with a five-dimensional interpolation table (results library).

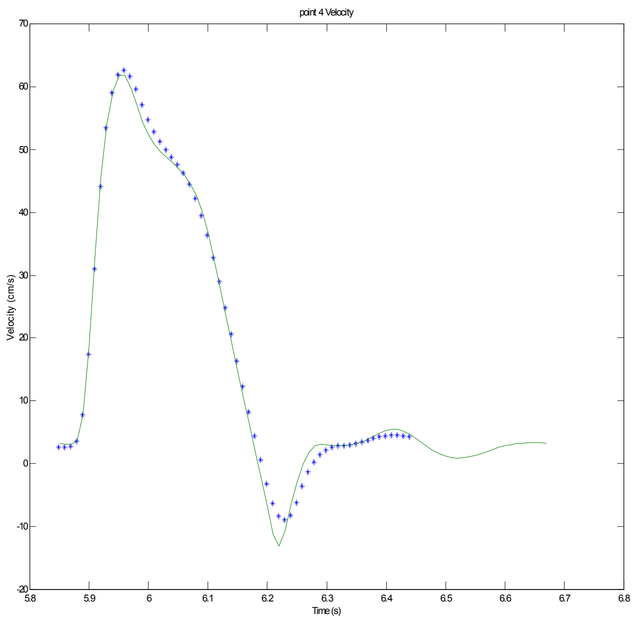


Fig. 6: Comparison of a typical result from the interpolation with that from the calculation of the original model

Fig. 6 illustrates that the results from 5-D linear interpolation has good accuracy although the error is about five percent. It is good enough for most clinical and teaching purposes. There are several ways that one could increase the accuracy:

- Increase the number of tabulated values in the library;
- Smooth each curve of one set of variables in the library first, but this will increase the time of computing;
- Use nonlinear interpolation.

4. CONCLUSIONS

CORBA is an open standard for distributed technology. It is therefore not surprising to see Web-based healthcare and physiological applications using CORBA as the underlying technology. Automated memory management, built-in thread support, and applet mobility make Java an ideal choice to implement CORBA clients and servers. With the backing of major players like Sun Microsystems, Netscape, and Oracle, the combined use of Java and CORBA will become commonplace in healthcare and biology, even in enterprise systems. Because of wide support for IIOP in firewall proxies, the availability of free or low cost ORB implementations, and built-in support for IIOP in server-side Web applications such as Web servers, however, IIOP may not become the standard Internet protocol. [3]

XML retains the key SGML advantages of extensibility, structure and error checking, and has been designed for maximum expressive power, a minimal learning curve, and maximum ease of implementation. Being intended for the storage and manipulation of text making up humane-readable documents like Web pages, it is already used in many fields.

A composite solution using CORBA/IIOP, Java, (JDBC) and XML is replicable with many different types of physiological models that produce a variety of results: graphical, numerical, and images. This concept and the underlying software should be reusable by others. It is a good means of delivering physiological models that allows interactive and distributed use over the Internet. The IIOP architecture can be considered as a precursor to a large Physiome project [5] with much more complex distributed models.

REFERENCES

[1] C. Forbes Dewey, Jr. [1], Ben Fu [1], Shixin Zhang [1], Ngon Dao [1], William Chuang [1] and Zheng LI [2], "An Information Architecture for Physiological Models, Clients and Databases," paper submited to *23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* Istanbul, Turkey, 25-28 October 2001.
[2] Xinshu Xiao, "Noninvasive Assessment of Cardiovascular Health," the thesis of the Degree of Master of Science in Mechanical Engineering of MIT, 1 August 2000.
[3] Reaz Hoque, *CORBA 3,* ISBN 0-7645-3200-6, IDG Books Worldwide, Inc, An International Data Group Company, pp. 21-27, 347-352, 402, 1998.
[4] Java, RMI and CORBA, *http://www. omg. org/library/wpjava. html.*
[5] Bassingthwaighte, J. B., "Strategies for the physiome project," *Annals of Biomedical Engineering,* vol. 28, pp. 1043-1058, 2000.